Protocol Binding Templates

September 20, 2017

Outline and Examples

Use cases

- Mapping of the WoT abstract interactions (Events, Actions, Properties) to concrete protocols
- Devices and web services modeled using TD interactions
- Existing device ecosystems like OCF, LWM2M/IPSO, Bluetooth, Zigbee, Echonet
- HTTP, CoAP, MQTT, Websocket protocols
- IP and non-IP transports
- Multi-servient synchronization, protocol adaptation and model re-composition

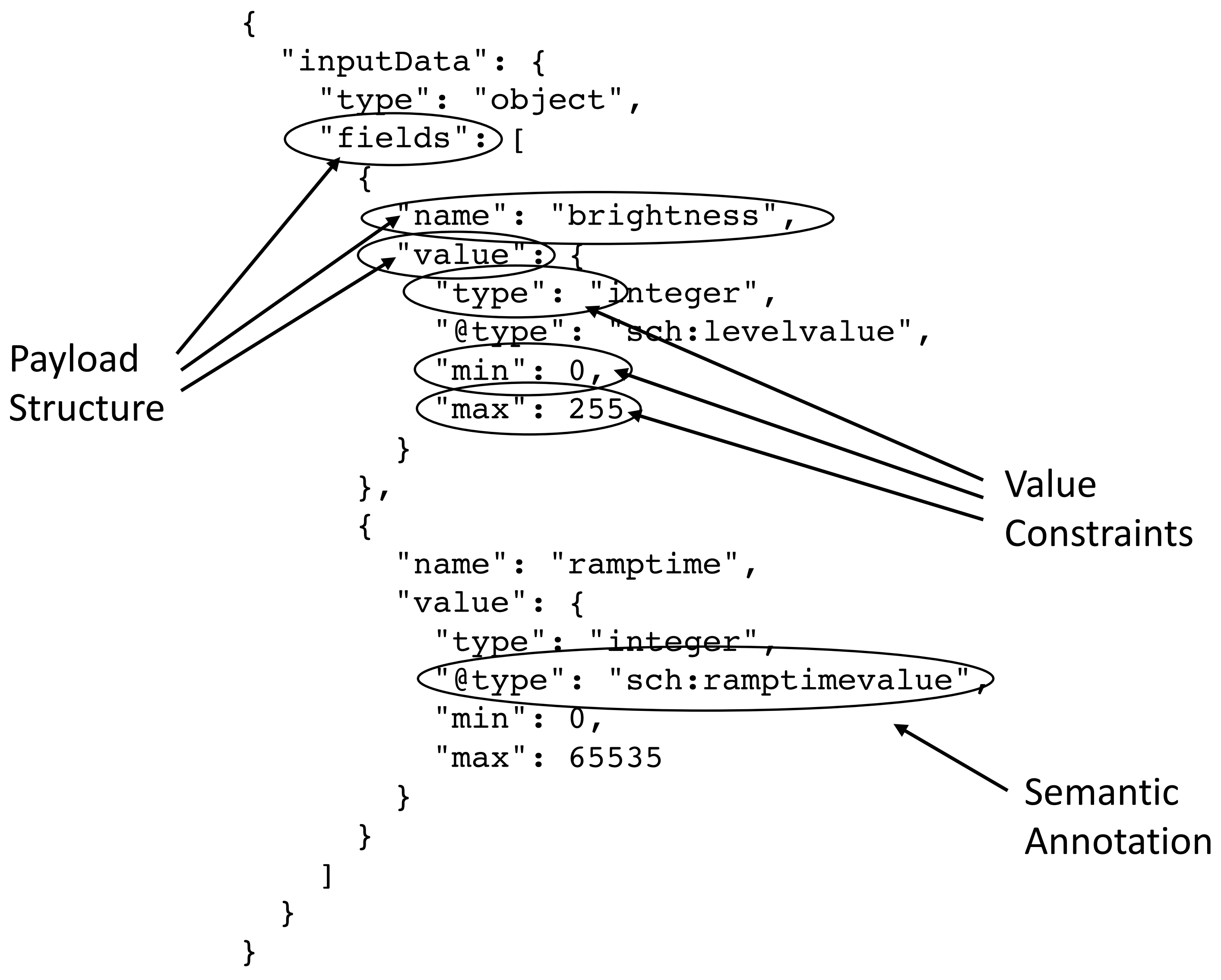
Protocol Adaptation

- The client uses information in the Thing Description to adapt to protocol specifics
  - Payload Structure
  - Value Constraints
  - Protocol Address
  - Protocol Methods
  - Protocol Options
  - Serialization Type (internet media type)

TD Interaction Design Pattern

- InputData and outputData elements describe the payload structural format
- Value Constraints enable the client to scale
- Protocol Address, method, options, and mediatype are described in the link element
- Semantic Annotation identifies the purpose of individual items in the data elements

Example inputData Element



```
{
  "inputData": {
    "type": "object",
    "fields": [
      {
        "name": "brightness",
        "value": {
          "type": "integer",
          "@type": "sch:levelvalue",
          "min": 0,
          "max": 255
        }
      },
      {
        "name": "ramptime",
        "value": {
          "type": "integer",
          "@type": "sch:ramptimevalue",
          "min": 0,
          "max": 65535
        }
      }
    ]
  }
}
```

Payload Structure

```
{
  "brightness": 127,
  "ramptime": 100
}
```

Payload Variation by Protocol

OCF Batch Interface

```
[
  {
    "href": "brightness",
    "rep": {
      "brightness:": 50
    }
  },
  {
    "href": "ramptime",
    "rep": {
      "ramptime:": 10000
    }
  }
]
```

LWM2M/IPSO

```
{
  "bn": "/3001/0/",
  "e": [
    {
      "n": "5001",
      "v": 0.5
    },
    {
      "n": "5003",
      "v": 10.0
    }
  ]
}
```

inputData for OCF Batch

```
{
  "inputData": {
    "type": "array",
    "items": [
      {
        "type": "object",
        "fields": [
          {
            "name": "href",
            "value": "brightness"
          },
          {
            "name": "rep",
            "type": "object",
            "fields": [
              {
                "name": "brightness",
                "value": {
                  "type": "integer",
                  "@type": "sch:levelvalue",
                  "min": 0,
                  "max": 100
                }
              }
            ]
```

inputData for LWM2M/IPSO

```
{
  "inputData": {
    "type": "object",
    "fields": [
      {
        "name": "bn",
        "value": "/3001/0/"
      },
      {
        "name": "e",
        "type": "array",
        "items": [
          {
            "type": "object",
            "fields": [
              {
                "name": "n",
                "value": "5001"
              },
              {
                "name": "v",
                "value": {
                  "type": "float",
                  "@type": "sch:levelvalue",
                  "min": 0.0,
                  "max": 1.0
                }
              }
```

Link Metadata Example

```
"link": [
  {
    "href": "/light",
    "mediatype": "application/vnd.ocf+cbor",
    "method": "post",
    "queryoptions": {
      "rt": ["oic.r.brightness", "oic.r.ramptime"],
      "if": ["oic.if.b"],
    },
    "headeroptions": {
      "12": 10000,
      "2053": 2048
    }
  }
]
```

Bindings for Properties

- Properties have get and set operations, the binding provides a way to specify the method in the target protocol that is to be used for the property.set operation

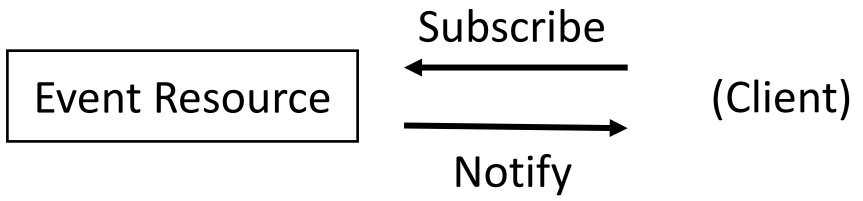
Bindings for Actions

- Actions have an invoke method on the target of the link, the method metadata defines which method in the target protocol is to be used for action.invoke
- action.invoke may create an action resource and return a TD describing the created resource
- The created resource has methods for updating and deleting the action resource as a way to modify the execution of the action in progress

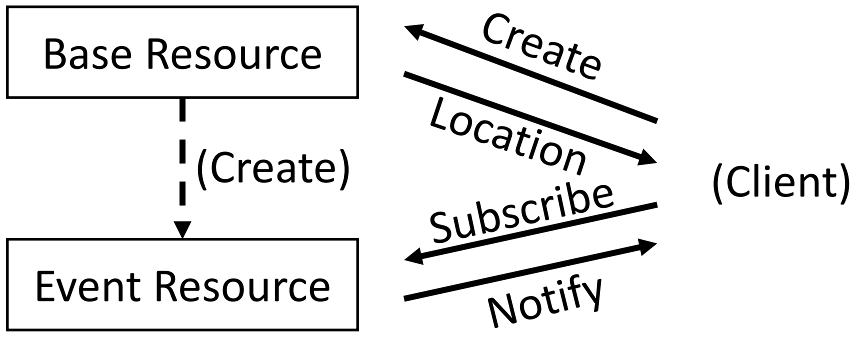
Bindings for Events

- The Event binding describes how a client can monitor a source of events
- The client may use, for example, CoAP Observe, MQTT Subscribe, HTTP EventSource, or Websockets
- Events may use a create-subscription pattern to obtain a TD for a newly created resource which then may be monitored according to the returned TD
- A CoAP Observable resource may be a source of events

Event Patterns



This pattern is used where there is a pre-configured event source. for example CoAP Observable resources or fixed Pub/Sub Topics



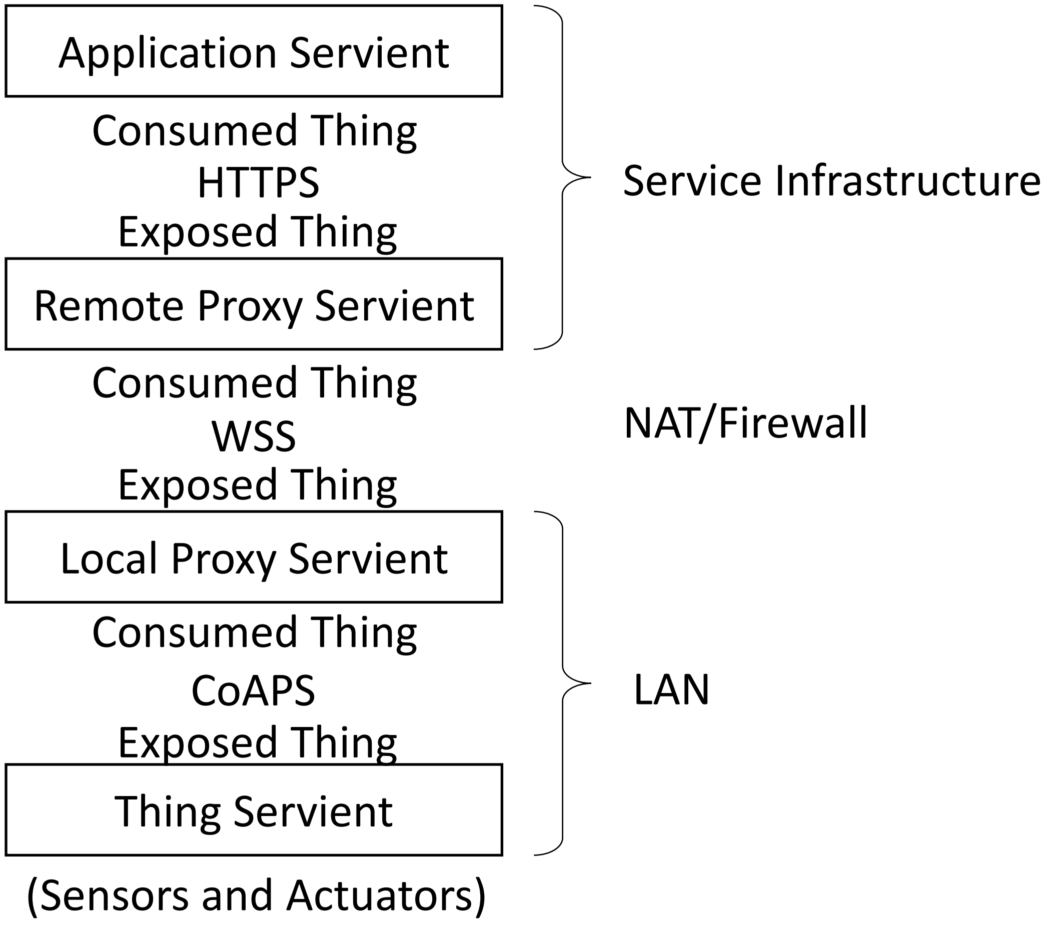
This pattern is used when event sources are configured dynamically, for example websocket events or dynamic Pub/Sub Topics

- We need to indicate which pattern the Event resource exposes

Vocabulary for Binding Templates

- Methods:
  - "method" keyword
  - "get", "put", "post", "delete", "patch", "subscribe", "observe", "publish" allowed values
- Options:
  - "headeroptions", "queryoptions" map keywords
  - key-value maps according to the concrete protocol
- Action keywords:
  - keywords for update and delete action operations?
- Event Keywords:
  - inputData parameters for conditional notification ?

Bindings for Proxy Servients


Application Servient
Consumed Thing
HTTPS
Exposed Thing
Remote Proxy Servient
Consumed Thing
WSS
Exposed Thing
Local Proxy Servient
Consumed Thing
CoAPS
Exposed Thing
Thing Servient
(Sensors and Actuators)
Service Infrastructure
NAT/Firewall
LAN